HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200311053-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Christopher P. Ruemmler, et al.

Confirmation No.: 4526

Application No.: 10/670,026

Examiner: Kawsar, Abdullah AL

Filing Date: 09/24/2003

Group Art Unit: 2195

Title: REDUCING LATENCY WHEN ACCESSING TASK PRIORITY LEVELS

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on 02/08/2008.

☒ The fee for filing this Appeal Brief is $510.00 (37 CFR 41.20).

☐ No Additional Fee Required.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

☐ (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d)) for the total number of months checked below:

☐ 1st Month $120	☐ 2nd Month $460	☐ 3rd Month $1050	☐ 4th Month $1640

☐ The extension fee has already been filed in this application.

☒ (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account 08-2025 the sum of $ 510. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees.

☒ A duplicate copy of this transmittal letter is enclosed.

☒ I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to:
Commissioner for Patents, Alexandria, VA 22313-1450
Date of Deposit: March 25, 2008

OR

☐ I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571)273-8300.
Date of facsimile:

Typed Name: James K. Okamoto
Signature: _____

Respectfully submitted,

Christopher P. Ruemmler, et al.

By _____

James K. Okamoto

Attorney/Agent for Applicant(s)

Reg No. : 40,110

Date : March 25, 2008

Telephone : (408) 436-2110

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Examiner: Kawsar, Abdullah Al

Christopher Philip Ruemmler et al.

Application No. 10/670,026 Art Unit: 2109

Filing Date: September 24, 2003 Attorney Docket No.: 200311053-1

Title: Reducing Latency When Accessing Task Priority Levels

Honorable Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF FILED UNDER 37 C.F.R. § 41.37

Sir:

This appeal brief follows the Notice of Appeal filed by Applicants on February 8, 2008.

The Commissioner is hereby authorized to charge requisite fees due for this submission to Deposit Account No. 08-2025 (Hewlett Packard).

I. REAL PARTY IN INTEREST

The real party in interest is the Hewlett-Packard Development Company, L.P., a Texas Limited Partnership having its principal place of business in Houston, Texas. The Hewlett-Packard Development Company, L.P., is the assignee of the present application.

II. RELATED APPEALS AND INTERFERENCES

On information and belief, there are no appeals, interferences, or judicial proceedings known to the appellant, the appellant's legal representative, or assignee which may be related to, directly affect or be directly affected by or have a bearing on the Board of Patent Appeals and Interferences (the "Board") decision in the pending appeal.

III. STATUS OF CLAIMS

A. Total Claims: 1-29

B. Current Status of Claims:

1. Claims canceled: 22-25
2. Claims withdrawn: none
3. Claims pending: 1-21 and 26-29
4. Claims allowed: none
5. Claims rejected: 1-21 and 26-29
6. Claims objected to: none

C. Claims on Appeal: 1-21 and 26-29

IV. STATUS OF AMENDMENTS

No amendment has been filed after the final rejection mailed November 9, 2007.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The claimed subject matter relates to microprocessors and operating systems. More particularly, the claimed subject matter relates to interrupt masks and creating and maintaining a shadow copy of a task priority register. Advantages include reduced latencies for writing to and reading from the task priority register.

Independent claim 1 relates to a method of reducing access latency to a task priority register (TPR) of a local programmable interrupt controller unit within a microprocessor. (Specification, FIG. 5, page 8, lines 4-23.) A command is received to write an interrupt mask value to the TPR, and the interrupt mask value is written to the TPR. (Specification, receive command 302 and write interrupt mask to TPR 304 in FIG. 5, page 8 lines 6-9.) In addition, the interrupt mask value is also written into a shadow copy of the TPR. (Specification, write interrupt mask to shadow copy of TPR 502 in FIG. 5, page 8, lines 9-12.) The shadow copy is written each time that the TPR is written. (Specification, blocks 304 and 502 and arrow therebetween in FIG. 5, page 8, lines 6-12.)

Independent claim 14 relates to a computer-readable medium storing an operating system with reduced access latency to a task priority register of a local programmable interrupt controller unit within a microprocessor. (Specification, FIG. 5, page 8, lines 4-23.) Microprocessor-executable code is configured to write a priority level to the task priority register. (Specification, write interrupt mask to TPR 304 in FIG. 5, page 8, lines 8-9.) In addition, microprocessor-executable code is also configured to write the priority level into a shadow copy of the task priority register. (Specification, write interrupt mask to shadow copy of TPR 502 in FIG. 5, page 8, lines 9-12.) The shadow copy is written each time that the task priority register is written. (Specification, blocks 304 and 502 and arrow therebetween in FIG. 5, page 8, lines 6-12.)

Independent claim 26 relates to a method of reducing a latency to read a TPR of a microprocessor. (Specification, FIG. 6, page 8, line 24 through page 9, line 5.) When a command is received to read an interrupt mask value from the TPR (Specification, receive command to read interrupt mask from TPR 402 in FIG. 6, page 8, lines 25-26), the interrupt mask value is read from the shadow copy at a memory location, instead of from the task priority register itself. (Specification, read interrupt mask from shadow copy 602 in FIG. 6, page 8, line 26 through page 9, line 5. Contrast with read interrupt mask from TPR 404 in FIG. 4.)

Independent claim 27 relates to an operating system stored on a computer-readable medium with reduced access latency to a task priority register of a local programmable interrupt controller unit within a microprocessor. (Specification, FIG. 5,

page 8, lines 4-23.) Microprocessor-executable code is configured to write a priority level to the task priority register. (Specification, write interrupt mask to TPR 302 in FIG. 5, page 8 lines 6-7.) In addition, microprocessor-executable code is also configured to write the priority level into a shadow copy of the task priority register. (Specification, write interrupt mask to shadow copy of TPR 502 in FIG. 5, page 8, lines 9-23.) The shadow copy is written each time that the task priority register is written. (Specification, blocks 304 and 502 and arrow therebetween in FIG. 5, page 8, lines 6-12.)

Independent claim 28 relates to a multiple-processor computer system. (Specification, FIG. 9, page 10, line 21 through page 11, line 4.) Multiple microprocessors are interconnected by a processor bus. (Specification, multiple processors 902 and processor bus 906 in FIG. 9.) Each microprocessor includes a task priority register (TPR) with a interrupt mask value for that microprocessor. (Specification, TPR 904 is shown in each microprocessor 902 in FIG. 9.) A memory system, including local cache memory on each microprocessor and a main memory, holds data including an operating system and shadow copies of the TPRs. (Specification, local memories 910 and shadow copies 912 in FIG. 9.) The operating system includes executable-code for reading the interrupt mask values from the shadow copies and for maintaining the shadow copies. (Specification, reading the interrupt mask value from the shadow copy is shown in block 602 of FIG. 6, and writing the interrupt mask to the shadow copy is shown in block 502 of FIG. 5.)

Independent claim 29 relates to a method of reducing latency to write a task priority register within a microprocessor. (Specification, FIG. 5, page 8, lines 4-23.) Upon receiving a command to write an interrupt mask value to the task priority register, the interrupt mask value is written to the task priority register without performing a serialization directly thereafter. (Specification, receiving command to write an interrupt mask to TPR 302, write interrupt mask to TPR 304 in FIG. 5, page 8 lines 6-7. Contrast with write interrupt mask to TPR 304 and serialize 306 in FIG. 3.) Upon receiving an interrupt, the serialization and reading an interrupt vector register are performed, wherein a spurious indicator is returned if the interrupt is maskable. (Specification, receiving interrupt 702, serialize 704, and read interrupt mask from TPR 802 in FIG. 8, spurious vector returned 250 in FIG. 2.)

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The following grounds of rejection are to be reviewed on appeal:

1. The rejection of claims 1-4, 6, 8-17, 19, 21 and 26-29 under 35 U.S.C. § 103(a) as being unpatentable over US Patent No. 6,665,750 (Williams) in view of "Intel Itanium Processor Family Interrupt Architecture Guide" (Intel).

2. The rejection of claims 5, 7, 18 and 20 under 35 U.S.C. § 103(a) as being unpatentable over US Patent No. 6,665,750 (Williams) in view of "Intel Itanium Processor Family Interrupt Architecture Guide" (Intel), and further in view of US Patent No. 7,080,205 (Demharter).

VII. ARGUMENT

Applicants respectfully traverse the aforementioned rejections of claims 1-21 and 26-29 in the last office action for the following reasons.

Technological Distinction between Interrupt Requests and Interrupt Masks

For preliminary purposes of explaining the relevant background technology, the following discussion explains the differences in function and use between an **interrupt request** typically stored in an interrupt register and an **interrupt mask** typically stored in a priority register.

As is known in the microprocessor art, an **interrupt request** (also called an interrupt vector, interrupt signal, or simply an interrupt) may come from various sources. For example, each keystroke typically generates an interrupt request. An interrupt request is typically stored in an **interrupt register**. A general definition for such an interrupt is given, for example, by www.webopedia.com as follows.

> **Interrupt**
> **(n.)** A signal informing a program that an event has occurred. When a program receives an interrupt signal, it takes a specified action (which can be to ignore the signal). Interrupt signals can cause a program to suspend itself temporarily to service the interrupt.

> Interrupt signals can come from a variety of sources. For example, every keystroke generates an interrupt signal. Interrupts can also be generated by other devices, such as a printer, to indicate that some event has occurred. These are called *hardware interrupts.* Interrupt signals initiated by programs are called *software interrupts*. A software interrupt is also called a *trap* or an *exception.*
>
> PCs support 256 types of software interrupts and 15 hardware interrupts. Each type of software interrupt is associated with an *interrupt handler* -- a routine that takes control when the interrupt occurs. For example, when you press a key on your keyboard, this triggers a specific interrupt handler. The complete list of interrupts and associated interrupt handlers is stored in a table called the *interrupt vector table*, which resides in the first 1 K of addressable memory.
> Also see the list of IRQ numbers in the Quick Reference section of Webopedia.

(Retrieved from http://www.webopedia.com/ on August 23, 2007.)

In contrast, an **interrupt mask** is not an interrupt request. An interrupt mask is a mask which is **applied to interrupt requests so as to determine whether or not that interrupt request is to be processed or not**. For example, the definition of "interrupt mask" retrieved from www.pcmag.com on August 23, 2007 states as follows. "An internal switch setting that controls whether an interrupt can be processed or not."

As a further example, the background section of U.S. Patent No. 6,175,890, explains the difference between **interrupt requests** and **interrupt masks** as follows.

> In a system where a microprocessor is connected to a plurality of external devices, a plurality of **interrupt requests** for accessing the microprocessor may occur simultaneously. These requests may or may not be accepted depending on how the microprocessor is executing a current process.
>
> Acceptance of requests is controlled by an **interruption mask** flag (hereinafter, simply referred to as a mask flag) set in a mask register usually provided in the microprocessor. For example, an interruption is enabled when the mask flag is set to 0 and is disabled when the mask flag is set to 1. Hereinafter, a "mask level state" will refer to a status of a microprocessor process characterized by a unique set of mask flags.
>
> (Column 1, lines 13-25.)

As discussed above, interrupt requests and interrupt masks are entirely different in function and use.

A. Claims 1, 3-4, 9-14, and 16-17

Claims 1, 3-4, 9-14, and 16-17 stand rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,665,750 (Williams) in view of "Intel Itanium Processor Family Interrupt Architecture Guide" (Intel). Applicants respectfully traverse this rejection.

In order to establish a prima facie case of obviousness, the prior art reference or combined references must teach or suggest all the claim limitations.

Claim 1 recites as follows.

> 1. A method of reducing access latency to a task priority register of a local programmable interrupt controller unit within a microprocessor, the method comprising:
> receiving a command to write an interrupt mask value to the task priority register;
> writing the interrupt mask value to the task priority register; and
> writing the **interrupt mask** value into a **shadow copy of the task priority register,**
> wherein the shadow copy is written each time that the task priority register is written.
>
> (Emphasis added.)

As seen above, claim 1 recites "writing the **interrupt mask** value into a **shadow copy of the task priority register,** wherein the shadow copy is written each time that the task priority register is written." (Emphasis added.) Applicants respectfully submit that neither Williams, nor Intel, nor the combination thereof teaches or suggests writing the **interrupt mask** value into a **shadow copy of the TPR** each time that the TPR is written.

The latest office action asserts that the above-discussed claim limitation is unpatentable over column 3, lines 12-14 ("copying the internal status value to a prescribed location in the system memory based on detecting at least one interrupt condition") of Williams in view of the teaching of Intel (writing an interrupt mask to the task priority register). Applicants respectfully traverse the assertion.

Regarding Williams, applicants respectfully submit that the "interrupt status value" of Williams is stored in the interrupt register (see Abstract) and so is an **interrupt request, <u>not</u> an interrupt mask**. As explained above, an interrupt request stored in an interrupt register is entirely different in function and use compared with an interrupt mask

stored in a priority register. Hence, the disclosure of Williams of copying an **interrupt request** to system memory is **technologically distinct in function and use** compared with writing the **interrupt mask** value into a **shadow copy of the TPR** each time that the TPR is written. As explained above, the function of interrupt requests and interrupt masks are entirely different. Hence, applicants respectfully submit that there is no teaching or suggestion in Williams of writing the **interrupt mask** value into a **shadow copy of the TPR** each time that the TPR is written.

Regarding Intel, the disclosure of Intel of writing an interrupt mask to the task priority register relates to the claim limitations of "receiving a command to write an interrupt mask value to the task priority register" and "writing the interrupt mask value to the task priority register". However, applicants respectfully submit that there is no teaching or suggestion in Intel of writing the **interrupt mask** value into a **shadow copy of the TPR** each time that the TPR is written.

Therefore, applicants respectfully submit that claim 1 overcomes its rejection.

Claims 3-4 and 9-13 depend from claim 1. Thus, claims 3-4 and 9-13 are patentable over Williams in view of Intel for at least the reasons discussed above in relation to claim 1.

Claim 14 is a computer-readable medium claim which recites limitations similar to the limitations of claim 1. Thus, claim 14 is patentable over Williams in view of Intel for at least the reasons discussed above in relation to claim 1.

Claims 16-17 depend from claim 14. Thus, claims 16-17 are patentable over Williams in view of Intel for at least the reasons discussed above in relation to claim 1.

B. Claims 26-28

Claims 26-28 stand rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,665,750 (Williams) in view of "Intel Itanium Processor Family Interrupt Architecture Guide" (Intel). Applicants respectfully traverse this rejection.

In order to establish a prima facie case of obviousness, the prior art reference or combined references must teach or suggest all the claim limitations.

Claim 26 recites as follows.

26. A method of reducing a latency to read a task priority register of an IPF type microprocessor, the method comprising:
receiving a command to read an interrupt mask value from the task priority register; and
reading the interrupt mask value from the shadow copy at a memory location, **instead of from the task priority register itself.**

(Emphasis added.)

As seen above, claim 26 recites "**reading the interrupt mask value from the shadow copy** at a memory location, **instead of from the task priority register itself**," in response to a command to read the mask from the TPR. (Emphasis added.) Applicants respectfully submit that neither Williams, nor Intel, nor the combination thereof teaches or suggests **reading the interrupt mask value from a shadow copy** of the TPR, **instead of from the TPR itself**, in response to a command to read the mask from the TPR.

The latest office action asserts that the above-discussed claim limitation is unpatentable over column 4, lines 34-36 ("enabling the CPU 12 to read the interrupt status value from the system memory location 30a in response to an interrupt") of Williams in view of the teaching of Intel (writing an interrupt mask to the task priority register). Applicants respectfully traverse the assertion.

Regarding Williams, applicants respectfully submit that there is no teaching or suggestion in Williams of **reading an interrupt mask** value from a **shadow copy of the TPR**, instead of from the TPR itself, in response to a command to read the mask from the TPR. The "interrupt status value" of Williams is stored in the interrupt register (see Abstract) and so is an **interrupt request** (interrupt vector), **not an interrupt mask**. As explained above, an interrupt request is technologically different in function and use compared with an interrupt mask. Hence, the disclosure of Williams of **reading an interrupt request** from system memory in response to an interrupt is **technologically distinct in function and use** compared with **reading an interrupt mask** value from a **shadow copy of the TPR**, instead of from the TPR itself, in response to a command to read the mask from the TPR.

Regarding Intel, the disclosure of Intel of writing an interrupt mask to the task priority register does not teach or suggest reading the **interrupt mask** value from a

shadow copy of the TPR, instead of from the TPR itself, in response to a command to read the mask from the TPR.

Therefore, applicants respectfully submit that claim 26 overcomes its rejection.

Claim 27 is a computer-readable medium claim which recites limitations similar to the limitations of claim 26. Thus, claim 27 is patentable over Williams in view of Intel for at least the reasons discussed above in relation to claim 26.

Claim 28 is a multiple-processor computer system claim which recites limitations similar to the limitations of claim 26. Thus, claim 28 is patentable over Williams in view of Intel for at least the reasons discussed above in relation to claim 26.

C. Claims 2 and 15

Claims 2 and 15 stand rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,665,750 (Williams) in view of "Intel Itanium Processor Family Interrupt Architecture Guide" (Intel). Applicants respectfully traverse this rejection.

In order to establish a prima facie case of obviousness, the prior art reference or combined references must teach or suggest all the claim limitations.

Claim 2 depends from claim 1 and recites as follows.

> 2. The method of claim 1, further comprising:
> receiving a command to read the interrupt mask value from the task priority register;
> reading the interrupt mask value from the shadow copy, instead of from the task priority register.
>
> (Emphasis added.)

As seen above, claim 2 recites "**reading the interrupt mask value from the shadow copy, instead of from the task priority register itself,**" in response to a command to read the mask from the TPR. (Emphasis added.) In addition, claim 2 depends from claim 1 and so also requires "writing the **interrupt mask** value into a **shadow copy of the task priority register,** wherein the shadow copy is written each time that the task priority register is written." (Emphasis added.) Applicants respectfully submit that

neither Williams, nor Intel, nor the combination thereof teaches or suggests the combined limitations of claims 1 and 2. The arguments for the limitations relating to writing and reading the interrupt mask value from a shadow copy are discussed above separately in sections A and B, respectively.

Therefore, applicants respectfully submit that claim 2 overcomes its rejection.

Claim 15 is a computer-readable medium claim which recites limitations similar to the limitations of claim 2. Thus, claim 15 is patentable over Williams in view of Intel for at least the reasons discussed above in relation to claim 2.

D. Claims 6, 19 and 29

Claims 6, 19 and 29 stand rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,665,750 (Williams) in view of "Intel Itanium Processor Family Interrupt Architecture Guide" (Intel). Applicants respectfully traverse this rejection.

In order to establish a prima facie case of obviousness, the prior art reference or combined references must teach or suggest all the claim limitations.

Claim 6 recites as follows.

> 6. The method of claim 1, wherein the method obviates a need to use a serialize instruction after the task priority register is written because **each interrupt performs a serialize operation** and performs a read of an interrupt vector register.
>
> (Emphasis added.)

As seen above, claim 6 recites that "the method obviates a need to use a serialize instruction after the task priority register is written because **each interrupt performs a serialize operation** and performs a read of an interrupt vector register." (Emphasis added.) Applicants respectfully submit that neither Williams, nor Intel, nor the combination thereof teaches or suggests this limitation.

The latest office action asserts that the above-discussed claim limitation is unpatentable over column 4, lines 31-34 ("copies the interrupt status values to the system memory location 30a specified by the system memory address register 36, enabling the

CPU 12 to read the interrupt status value from the system memory ...") of Williams in view of the teaching of Intel (writing an interrupt mask to the task priority register). Applicants respectfully traverse the assertion.

Regarding Williams, applicants respectfully submit that the cited portion of Williams does not disclose or suggest a serialize operation performed after each interrupt. In fact, **there is no mention or suggestion of a serialize operation in the citation**. As explained in the present application, "After the serialize instruction, the system is guaranteed to be in a consistent state such that reading values from registers will make sense." (Page 7, lines 18-20.) Regarding Intel, the disclosure of Intel of writing an interrupt mask to the task priority register also does not disclose or suggest a serialize operation performed after each interrupt.

Thus, the latest office action does not cite to any portion of any reference that discloses or suggests a serialize operation performed after each interrupt. Nor does the latest office action give any reasoning as to why this limitation would be obvious to one of ordinary skill in the art at the time of the invention.

Therefore, applicants respectfully submit that claim 6 overcomes its rejection.

Claim 19 is a computer-readable medium claim which recites limitations similar to the limitations of claim 6. Thus, claim 19 is patentable over Williams in view of Intel for at least the reasons discussed above in relation to claim 6.

Claim 29 is a method claim which recites limitations similar to the limitations of claim 6. Thus, claim 29 is patentable over Williams in view of Intel for at least the reasons discussed above in relation to claim 6.

E. Claims 7 and 20

Claims 7 and 20 stand rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,665,750 (Williams) in view of "Intel Itanium Processor Family Interrupt Architecture Guide" (Intel) further in view of US Patent No. 7,080,205 (Demharter). Applicants respectfully traverse this rejection.

In order to establish a prima facie case of obviousness, the prior art reference or combined references must teach or suggest all the claim limitations.

Claim 7 recites as follows.

7. The method of claim 6, whereby a **latency of writing** to the task priority register is substantially reduced.

(Emphasis added.)

As seen above, claim 7 recites "whereby a **latency of writing to the task priority register** is substantially reduced." Applicants respectfully submit that neither Williams, nor Intel, nor Demharter, nor the combination thereof teaches or suggests this limitation.

The latest office action cites column 2, lines 8-10 ("Storage of the interrupt vectors and interrupt handlers in a cache ensures significant reduction in **the processing time of an interrupt**.") (emphasis added) of Demharter against claim 7.

Applicant respectfully submits that "the processing time of an interrupt" per Demharter is **technologically distinct** from the claimed "latency of writing to the task priority register." **The processing time of an interrupt is the time it takes to process an interrupt request. The latency of writing to the task priority register is the time delay between the command to write the TPR and when the TPR is actually written.**

Therefore, applicants respectfully submit that claim 7 overcomes its rejection.

In addition, the Demharter citation relates to a reduction in processing time of an interrupt by **storing interrupt vectors and handlers in a cache**. In contrast, the reduced latency of writing the TPR per claim 7 depends from claim 6 which recites "**obviating a need to use a serialize instruction after the task priority register is written**" (emphasis added) as the source of the latency reduction. This is yet an additional reason why claim 7 overcomes its rejection.

Claim 20 is a computer-readable medium claim which recites limitations similar to the limitations of claim 7. Thus, claim 20 is patentable over Williams in view of Intel further in view of Demharter for at least the reasons discussed above in relation to claim 7.

F. Claims 5 and 18

Claims 5 and 18 stand rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,665,750 (Williams) in view of "Intel Itanium Processor Family Interrupt Architecture Guide" (Intel) further in view of US Patent No. 7,080,205 (Demharter). Applicants respectfully traverse this rejection.

In order to establish a prima facie case of obviousness, the prior art reference or combined references must teach or suggest all the claim limitations.

Claim 5 recites as follows.

> 5. The method of claim 1, wherein, if the task priority register is accessed frequently, then **the shadow copy is stored in low-latency cache memory** within the microprocessor.
>
> (Emphasis added.)

As seen above, claim 7 recites "wherein, if the task priority register is accessed frequently, then **the shadow copy is stored in low-latency cache memory** within the microprocessor." Applicants respectfully submit that neither Williams, nor Intel, nor Demharter, nor the combination thereof teaches or suggests this limitation.

The latest office action cites column 1, lines 66-67 ("This array makes it possible to use two separate cache memories, of which one, for example, is suitable for **storing the interrupt vectors and interrupt handlers** ....") (emphasis added) of Demharter against claim 5.

Applicants respectfully submit that **storing interrupt vectors and interrupt handlers** in cache memory per Demharter is **technologically distinct** from the claimed **storing of a shadow copy of the task priority register** in cache memory.

Therefore, applicants respectfully submit that claim 5 overcomes its rejection.

Claim 18 is a computer-readable medium claim which recites limitations similar to the limitations of claim 5. Thus, claim 18 is patentable over Williams in view of Intel further in view of Demharter for at least the reasons discussed above in relation to claim 5.

G. Claims 8 and 21

Claims 8 and 21 stand rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,665,750 (Williams) in view of "Intel Itanium Processor Family Interrupt Architecture Guide" (Intel). Applicants respectfully traverse this rejection.

In order to establish a prima facie case of obviousness, the prior art reference or combined references must teach or suggest all the claim limitations.

Claim 8 recites as follows.

> 8. The method of claim 2, whereby a **latency of reading from the task priority register** is substantially reduced.
>
> (Emphasis added.)

As seen above, claim 8 recites "whereby a **latency of reading from the task priority register** is substantially reduced." Applicants respectfully submit that neither Williams, nor Intel, nor the combination thereof teaches or suggests this limitation.

The latest office action cites column 4, lines 49-51 ("**minimizing 110 read operations** by the CPU 12, based on copying interrupt status values into the system memory 16") (emphasis added) of Williams against claim 8.

Applicant respectfully submits that minimizing read operations per Williams is **technologically distinct** from the claimed reducing a latency of reading from the TPR. **Minimizing read operations means that less read operations are performed. In contrast, reducing a latency of reading means that the delay between the read command and when the data is actually read is reduced.**

Therefore, applicants respectfully submit that claim 8 overcomes its rejection.

Claim 21 is a computer-readable medium claim which recites limitations similar to the limitations of claim 8. Thus, claim 21 is patentable over Williams in view of Intel further in view of Demharter for at least the reasons discussed above in relation to claim 8.

VIII. CONCLUSION

For at least the above reasons, applicants respectfully request that the rejections of claims 1-21 and 26-29 be reversed.

Respectfully submitted,
Christopher Philip Ruemmler et al

Dated: March 25, 2008

James K. Okamoto, Reg. No. 40,110
Okamoto & Benedicto LLP
P.O. Box 641330
San Jose, CA 95164
Tel.: (408)436-2110
Fax.: (408)436-2114

CERTIFICATE OF MAILING			
I hereby certify that this correspondence, including the enclosures identified herein, is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below. If the Express Mail Mailing Number is filled in below, then this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service pursuant to 37 CFR 1.10.			
Signature:			
Typed or Printed Name:	James K. Okamoto	Dated:	March 25, 2008
Express Mail Mailing Number (*optional*):			

CLAIMS APPENDIX

CLAIMS INVOLVED IN THE APPEAL

1. A method of reducing access latency to a task priority register of a local programmable interrupt controller unit within a microprocessor, the method comprising:

receiving a command to write an interrupt mask value to the task priority register;

writing the interrupt mask value to the task priority register; and

writing the interrupt mask value into a shadow copy of the task priority register,

wherein the shadow copy is written each time that the task priority register is written.

2. The method of claim 1, further comprising:

receiving a command to read the interrupt mask value from the task priority register; and

reading the interrupt mask value from the shadow copy, instead of from the task priority register.

3. The method of claim 1, wherein the shadow copy is always written after the task priority register is written.

4. The method of claim 3, further comprising, upon receiving an interrupt, reading the interrupt mask value from the task priority register and writing the interrupt mask value to the shadow copy.

5. The method of claim 1, wherein, if the task priority register is accessed frequently, then the shadow copy is stored in low-latency cache memory within the microprocessor.

6. The method of claim 1, wherein the method obviates a need to use a serialize instruction after the task priority register is written because each interrupt performs a serialize operation and performs a read of an interrupt vector register.

7. The method of claim 6, whereby a latency of writing to the task priority register is substantially reduced.

8. The method of claim 2, whereby a latency of reading from the task priority register is substantially reduced.

9. The method of claim 1, wherein data in the task priority register reflects a level of priority of tasks being performed by the microprocessor.

10. The method of claim 9, wherein the task priority register comprises eight bits to designate up to 256 priority states.

11. The method of claim 1, wherein the method is performed by an operating system.

12. The method of claim 4, further comprising, after writing the interrupt mask to the shadow copy, reading an interrupt vector register at a beginning of an interrupt handler.

13. The method of claim 12, further comprising executing instructions specific to the interrupt handler and returning from the interrupt handler.

14. A computer-readable medium storing an operating system with reduced access latency to a task priority register of a local programmable interrupt controller unit within a microprocessor, the computer-readable medium comprising:

microprocessor-executable code configured to write a priority level to the task priority register; and

microprocessor-executable code configured to write the priority level into a shadow copy of the task priority register,

wherein the shadow copy is written each time that the task priority register is written.

15. The computer-readable medium of claim 14, further comprising:

microprocessor-executable code configured to read the priority level from the shadow copy, instead of from the task priority register.

16. The computer-readable medium of claim 14, wherein the shadow copy is always written after the task priority register is written.

17. The computer-readable medium of claim 16, further comprising,

microprocessor-executable code configured to, upon receipt of an interrupt, read the priority level from the task priority register and write the priority level to the shadow copy.

18. The computer-readable medium of claim 14, wherein, if the task priority register is accessed frequently, then the shadow copy is stored in low-latency cache memory within the microprocessor.

19. The computer-readable medium of claim 14, wherein the operating system avoids a need to use a serialize instruction after the task priority register is written because each interrupt performs a serialize operation.

20. The computer-readable medium of claim 19, whereby a latency of writing to the task priority register is substantially reduced.

21. The computer-readable medium of claim 15, whereby a latency of reading from the task priority register is substantially reduced.

26. A method of reducing a latency to read a task priority register of a microprocessor, the method comprising:

receiving a command to read an interrupt mask value from the task priority register; and

reading the interrupt mask value from the shadow copy at a memory location, instead of from the task priority register itself.

27. An operating system stored on a computer-readable medium with reduced latency to read a task priority register of a local programmable interrupt controller unit within a microprocessor, the operating system comprising microprocessor-executable code configured read the interrupt mask value from the shadow copy at a memory location, instead of from the task priority register itself.

28. A multiple-processor computer system comprising:

a plurality of microprocessors interconnected by a processor bus, wherein each microprocessor includes a task priority register (TPR) with a interrupt mask value for that microprocessor;

a memory system, including local cache memory on each microprocessor and a main memory,

wherein the memory system holds data including an operating system and shadow copies of the TPRs, and

wherein the operating system includes executable-code for reading the interrupt mask values from the shadow copies and for maintaining the shadow copies.

29. A method of reducing latency to write a task priority register within a microprocessor, the method comprising:

upon receiving a command to write an interrupt mask value to the task priority register, writing the interrupt mask value to the task priority register without performing a serialization directly thereafter; and

upon receiving an interrupt, performing the serialization and reading an interrupt vector register, wherein a spurious indicator is returned if the interrupt is maskable.

EVIDENCE APPENDIX

There are no documents or items submitted under this section.

RELATED PROCEEDINGS APPENDIX

There are no documents or items submitted under this section.